BCD-to-Excess-3 Code converter

- ✓ BCD is a code for the decimal digits 0-9
- ✓ Excess-3 is also a code for the decimal digits

Decimal Digit	Input BCD	Output Excess-3
0	0 0 0 0	0 0 1 1
1	0 0 0 1	0 1 0 0
2	0 0 1 0	0 1 0 1
3	0 0 1 1	0 1 1 0
4	0 1 0 0	0 1 1 1
5	0 1 0 1	1 0 0 0
6	0 1 1 0	1 0 0 1
7	0 1 1 1	1 0 1 0
8	1 0 0 0	1 0 1 1
9	1 0 0 1	1 1 0 0

Specification of BCD-to-Excess3

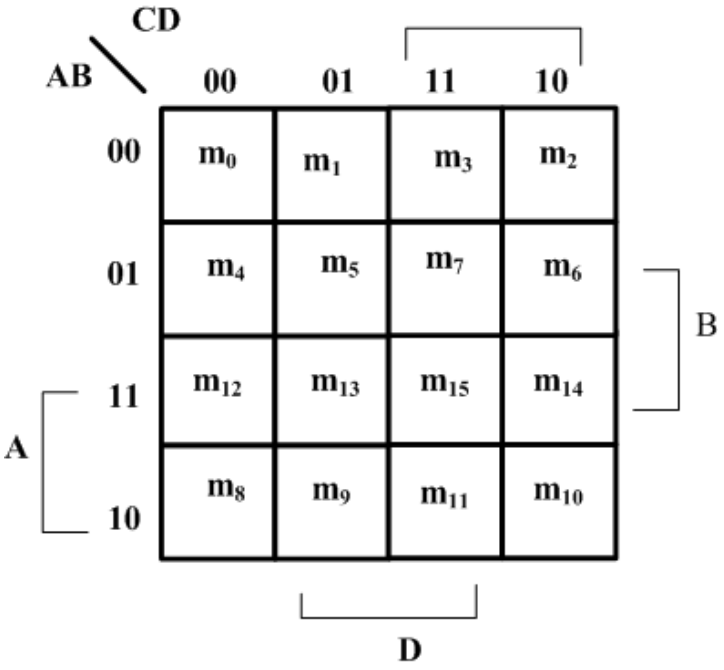
- ✓ Inputs: a BCD input, A,B,C,D with A as the most significant bit and D as the least significant bit.
- ✓ Outputs: an Excess-3 output W,X,Y,Z that corresponds to the BCD input.
- ✓ Internal operation – circuit to do the conversion in combinational logic.

Formulation of BCD-to-Excess-3

- ✓ Excess-3 code is easily formed by adding a binary 3 to the binary or BCD for the digit.
- ✓ There are 16 possible inputs for both BCD and Excess-3.
- ✓ It can be assumed that only valid BCD inputs will appear so the six combinations not used can be treated as don't cares.

Placing 1 on K-maps

- ✓ Where are the minterms located on a K-Map?



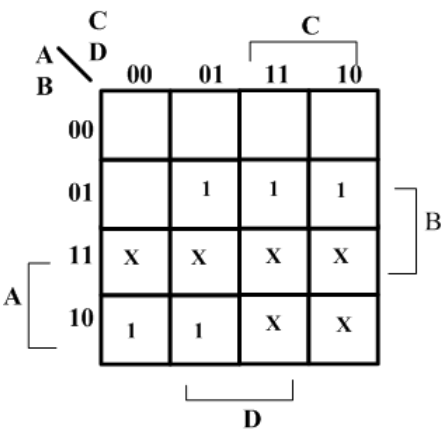
Expressions for W X Y Z

- ✓ $W(A,B,C,D) = \Sigma m(5,6,7,8,9) + d(10,11,12,13,14,15)$
- ✓ $X(A,B,C,D) = \Sigma m(1,2,3,4,9) + d(10,11,12,13,14,15)$
- ✓ $Y(A,B,C,D) = \Sigma m(0,3,4,7,8) + d(10,11,12,13,14,15)$
- ✓ $Z(A,B,C,D) = \Sigma m(0,2,4,6,8) + d(10,11,12,13,14,15)$

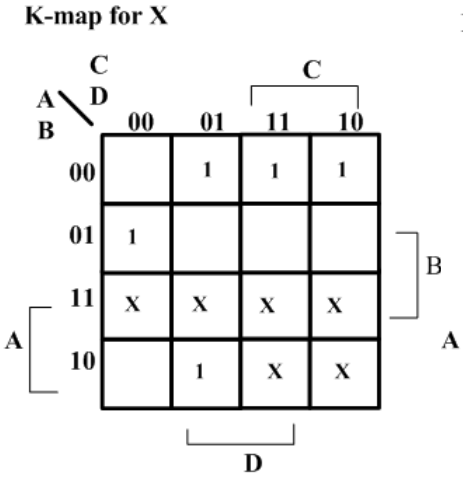
Optimization – BCD-to-Excess-3

- ✓ Lay out K-maps for each output, W X Y Z

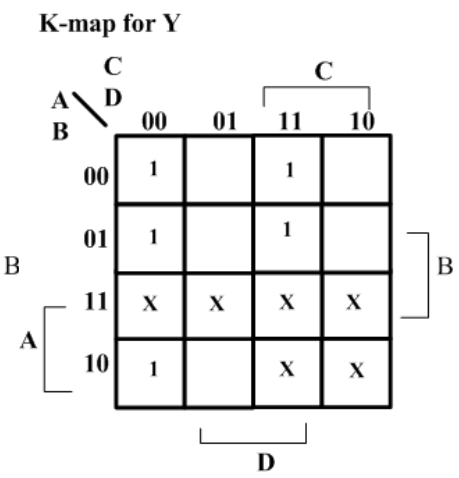
K-map for W



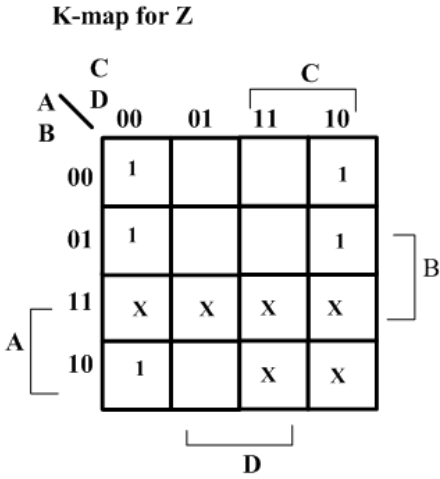
K-map for X



K-map for Y



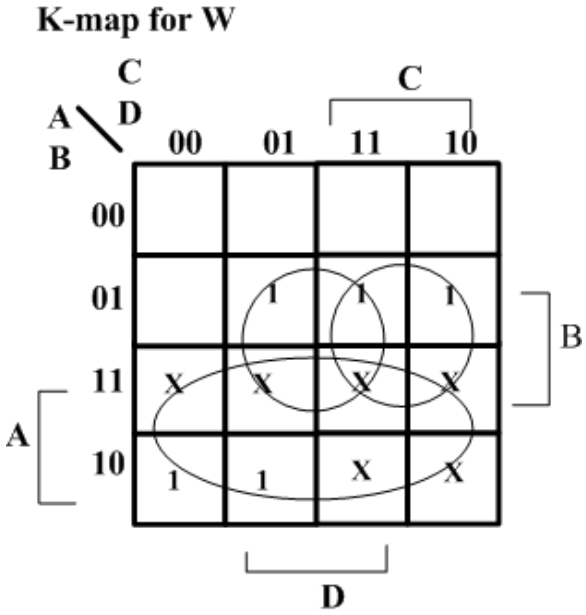
K-map for Z



- ✓ A step in the digital circuit design process.

Minimize K-Maps

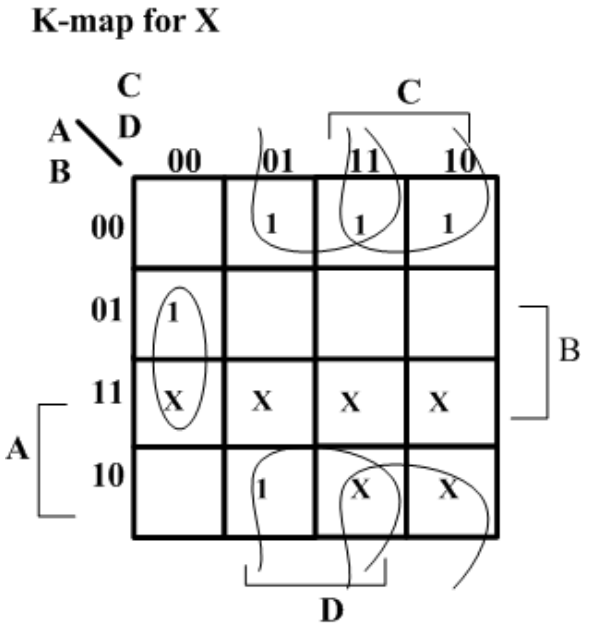
- ✓ W minimization



$W = A + BC + BD$

Minimize K-Maps

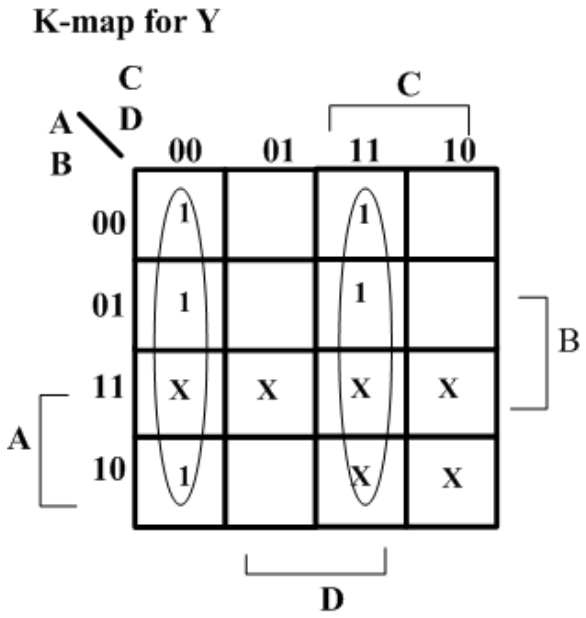
- ✓ X minimization



$X = BC'D' + B'C + B'D$

Minimize K-Maps

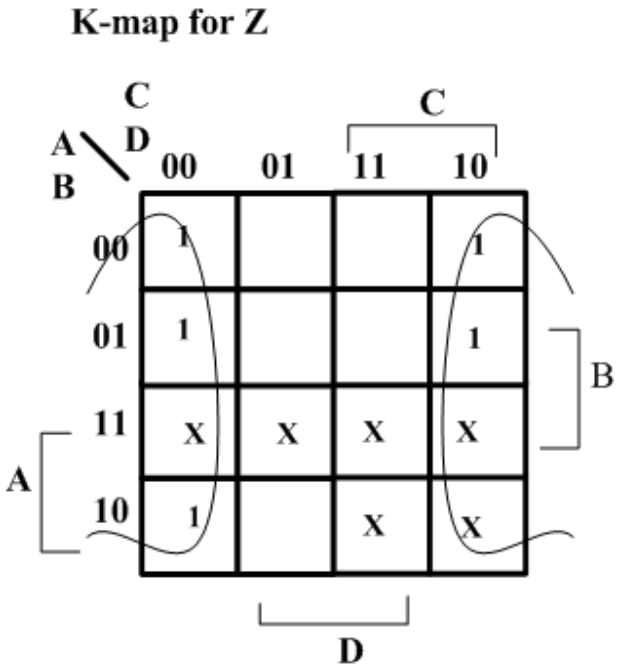
✓ Y minimization



$Y = CD + C'D'$

Minimize K-Maps

- ✓ Z minimization



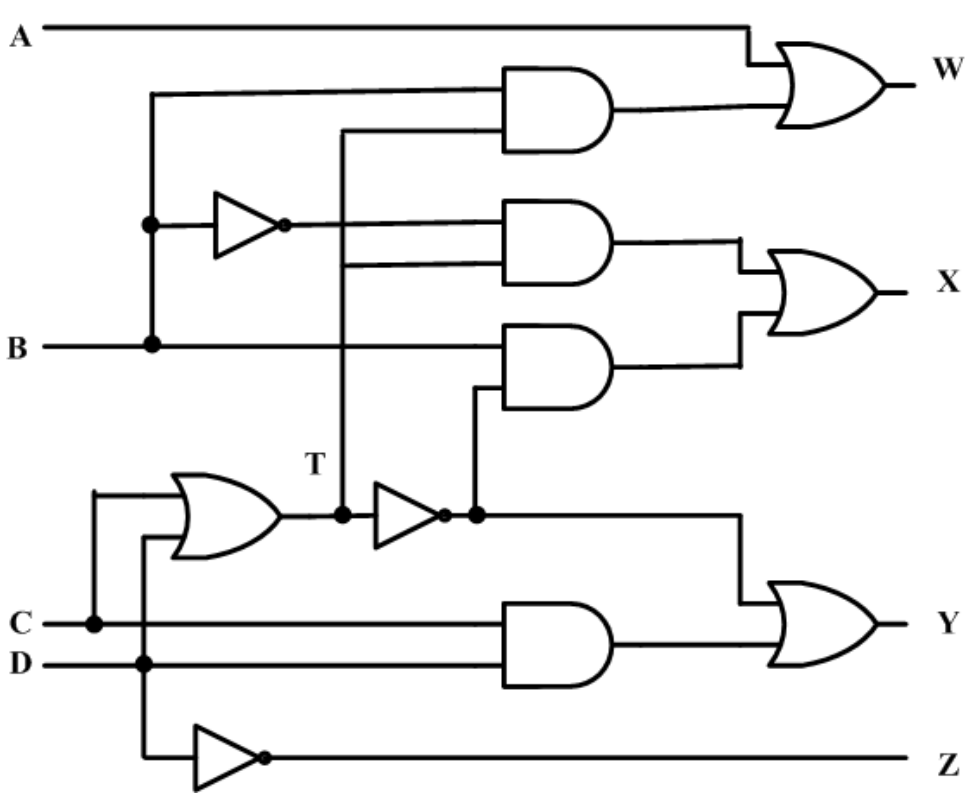
Z = D'

Two level circuit implementation

- ✓ Have equations
  - W = A + BC + BD = A + B(C+D)
  - X = B'C + B'D + BC'D' = B'(C+D) + BC'D'
  - Y = CD + C'D'
  - Z = D'
- ✓ Factoring out (C+D) and call it T
- ✓ Then T' = (C+D)' = C'D'
  - W = A + BT
  - X = B'T + BT'
  - Y = CD + T'
  - Z = D'

Create the digital circuit

- ✓ Implementing the second set of equations where T=C+D results in a lower gate count.
- ✓ This gate has a fanout of 3

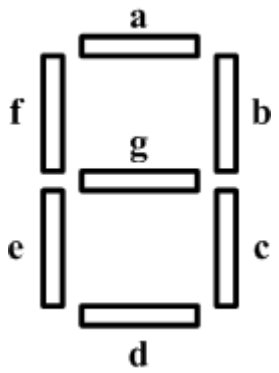


BCD-to-Seven-Segment Decoder

- ✓ Specification
  - Digital readouts on many digital products often use LED seven-segment displays.
  - Each digit is created by lighting the appropriate segments. The segments are labeled a,b,c,d,e,f,g
  - The decoder takes a BCD input and outputs the correct code for the seven-segment display.

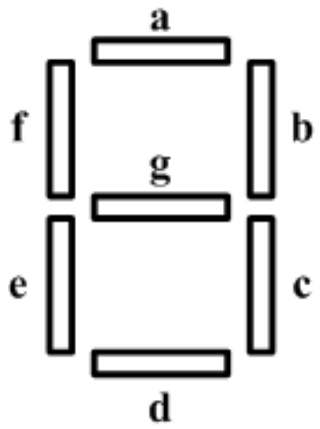
Specification

- ✓ Input: A 4-bit binary value that is a BCD coded input.
- ✓ Outputs: 7 bits, a through g for each of the segments of the display.
- ✓ Operation: Decode the input to activate the correct segments.



Formulation

- ✓ Construct a truth table



Decimal Digit	Input BCD	Seven-Segment Decoder Outputs a b c d e f g
0	0 0 0 0	1 1 1 1 1 1 0
1	0 0 0 1	0 1 1 0 0 0 0
2	0 0 1 0	1 1 0 1 1 0 1
3	0 0 1 1	1 1 1 1 0 0 1
4	0 1 0 0	1 0 1 1 0 1 1
5	0 1 0 1	1 0 1 1 0 1 1
6	0 1 1 0	1 0 1 1 1 1 1
7	0 1 1 1	1 1 1 0 0 0 0
8	1 0 0 0	1 1 1 1 1 1 1
9	1 0 0 1	1 1 1 1 0 1 1
All other inputs		0 0 0 0 0 0 0

Optimization

- ✓ Create a K-map for each output and get
  - A = A'C+A'BD+B'C'D'+AB'C'
  - B = A'B'+A'C'D'+A'CD+AB'C'
  - C = A'B+A'D+B'C'D'+AB'C'
  - D = A'CD'+A'B'C+B'C'D'+AB'C'+A'BC'D
  - E = A'CD'+B'C'D'
  - F = A'BC'+A'C'D'+A'BD'+AB'C'
  - G = A'CD'+A'B'C+A'BC'+AB'C'

Note on implementation

- ✓ Direct implementation would require 27 AND gates and 7 OR gates.
- ✓ By sharing terms, can actualize and implementation with 14 less gates maintaining just 2 levels.

- ✓ Normally decoder in a device name indicates that the number of outputs is less than the number of inputs.